ศึกษาภาษา โปรล็อก

บุญเลิศ เอี่ยมทัศนา

แนะนำตัว

เมื่อญี่ปุ่นประกาศว่า จะใช้ภาษาโปรล็อก กับเครื่องคอมพิวเตอร์ในยุคที่ห้าของตน ไม่มีใครทราบว่า ญี่ปุ่นกำลังทำอะไรอยู่ และไม่ทราบอีกเหมือนกันว่าคอมพิวเตอร์ในยุคที่ห้าจะมีหน้าตาเป็นอย่างไร แต่ก็ก่อให้เกิดการตื่นตัวในภาษานี้ขึ้นอย่างมาก ในสหรัฐอเมริกา จึงเป็นอีกภาษาหนึ่งที่เราน่าจะติดตามศึกษาไว้สำหรับอนาคต

ก่อนที่จะอ่านบทความของผม หากเป็นไปได้ขอให้อ่านเรื่อง "โปรล็อก:ภาษาอนาคต" ของอาจารย์สุรศักดิ์ สงวนพงษ์ ในไมโครฯ ฉบับที่ 18 เพื่อให้ทราบเรื่องราวโดยทั่วไปของภาษาโปรล็อก แต่เมื่ออ่านแล้วเห็นว่าโปรแกรมในฉบับนั้นกับฉบับนี้ต่างกัน ก็ขอเรียนให้ทราบว่า เป็นโปรล็อกเหมือนกัน แต่พัฒนาไปคนละแบบ แบบที่ผมนำมาใช้นี้เป็น PROLOG86 ของ MICRO-AI เหตุที่ผมเลือกใช้ก็เพราะมีคู่มือการใช้(User's guide and reference manual) และคิดว่าใกล้เคียงกับแบบมาตรฐาน เพราะใกล้เคียงกับที่กล่าวไว้ในหนังสือ Programming in Prolog ของ Clocksin และ Mellish ซึ่งน่าจะเรียกว่าเป็นคัมภีร์ภาษาโปรล็อก เพราะดูเหมือนจะเป็นหนังสือภาษาโปรล็อกเพียงเล่มเดียวที่ชอบอ้างถึงกัน

เก่ากับใหม่และตรรก

บางท่านกล่าวว่าภาษาโปรลือกเป็นภาษาใหม่ ความจริงภาษาโปรล็อกถือกำเนิดเกิดมาตั้งแต่พ.ศ.2513 เกิดก่อนภาษาที่ถูกเรียกว่าเก่าบางภาษาเสียอีก(เช่น ปาสคาล เกิดเมื่อปีพ.ศ.2514) บางท่านกล่าวว่าภาษาโปรลือกเป็นภาษาที่ใช้เขียนโปรแกรมเชิงตรรก คือนำมาจากชื่อของภาษาโปรลือกเอง (Prolog:Programming in Logic) ภาษานอกนั้นเป็นภาษาตามแบบ(conventional) ซึ่งเขียนโปรแกรมในแบบต่อเนื่อง(sequential)

แต่ความเป็นจริงแล้ว ภาษาตามแบบ (หากจะใช้คำนี้) เป็นภาษาที่เขียนโปรแกรมอย่างเป็นขั้นตอนไม่ใช่ต่อเนื่อง และเป็นวิธีการหนึ่งในเรื่องของตรรกวิทยา ที่เป็นเช่นนี้เพราะตรรกวิทยา เป็นวิชาที่ว่าด้วยเหตุและผล ไม่ว่าเหตุจะมาก่อนผลหรือผลจะมาก่อนเหตุ หลักก็มีอยู่เพียงเท่านี้ จึงสุดแต่ว่าจะนำไปประยุกต์ใช้ในทางใดที่เกี่ยวกับคอมพิวเตอร์ คือ การแก้ปัญหา ก็ต้องไปจับกันที่คำว่าปัญหา

กรรมวิธีในการแก้ปัญหา

มีหลักความจริงอยู่ว่า ปัญหาใดๆ ในโลกนี้ที่เราเห็นว่ามีมากมายนั้น ความจริงแล้วมีอยู่ 2 ลักษณะเท่านั้นคือ

– เป็นปัญหาข้อเท็จจริง เช่น ใช่หรือไม่ใช่ เชื่อได้หรือไม่ได้

– เป็นปัญหาเกี่ยวกับการเลือกหนทางปฏิบัติ เช่น มีเงินจะลงทุนทำอะไรดี (ซื้อทอง ซื้อพันธบัตร เล่นหุ้น ทำการค้า ฝากธนาคาร ฯลฯ)

ปัญหาทั้งสองลักษณะนี้ มีวิธีแก้ปัญหาอยู่หลายวิธี วิธีหนึ่งคือใช้กรรมวิธีทางตรรกเข้าแก้ปัญหา ซึ่งเป็นเรื่องที่ผมจะนำมากล่าวถึงต่อไปนี้

ปัญหาหนทางปฏิบัติ

มีหลักความจริงอยู่ว่า เมื่อมนุษย์เราประสบปัญหาเกี่ยวกับหนทางปฏิบัติ จะดำเนินการเป็น 3 ขั้นตอนต่อเนื่องกันไปโดยไม่รู้ตัวคือ คิด ตัดสินใจ และปฏิบัติ ซึ่งอาจจะใช้เวลาเป็น เสี้ยววินาที นาที ชั่วโมง วัน ปี หรือตลอดชีวิต(คือแก้ปัญหาไม่ตกเลยอำลาโลกไป) ทางตรรกจึงนำหลักความจริงนี้มาแสดงให้เห็น โดยแตกเป็นขั้นตอนย่อยดังแสดงในรูปที่ 1

จากรูปที่ 1 หากดูโดยผิวเผินจะเห็นเป็นว่า การแก้ปัญหากระทำต่อเนื่องจากข้อที่ 1 ถึงข้อที่ 8 แต่ความจริงแล้วการแก้ปัญหาจะทำเป็นขั้นตอนจากขั้นที่ 1 ถึงขั้นที่ 8 คือจะต้องนำผลจากขั้นตอนก่อนไปใช้ในขั้นตอนต่อไป และมีการวนรอบ เช่น หากกำหนดความมุ่งหมายไว้แคบเกินไป เมื่อมาถึงขั้นที่ 5 ไม่สามารถกำหนดหนทางปฏิบัติได้ จะต้องย้อนกลับไปกำหนดความมุ่งหมายใหม่

อีกประเด็นหนึ่งหากเป็นปัญหาที่กว้าง หนทางปฏิบัติที่เราเลือกแล้วอาจมีหลายหนทางปฏิบัติ เมื่อถึงขั้นปฏิบัติก็จะมีการปฏิบัติที่แตกแยกกันออกไป และในขั้นปฏิบัตินี้เองอาจมีการดำเนินกรรมวิธีตั้งแต่ขั้นคิดต่อไปใหม่ ดูเสมือนเป็นการแก้ปัญหาเป็นเปราะๆ ไป จึงถือว่าเป็นการแก้ปัญหาอย่างเป็นขั้นตอน(logical precess)

ปัญหาข้อเท็จจริง

ในกระบวนการของปัญหา ปัญหาที่เกี่ยวกับข้อเท็จจริงเป็นปัญหาที่แก้ยากที่สุด สมมติว่ามีเพื่อนมาชวนคุณว่า *"ไปตกปลากันเถอะวันนี้ฝนตกปลากินเบ็ดดี"* คุณจะเชื่อเขาหรือไม่ ประโยคนี้เป็นจริงหรือเท็จ หากนำกรรมวิธีทางตรรกเข้าจับ ประโยคนี้จะเป็นเท็จทันที

ความเป็นเท็จไม่ได้อยู่ที่วันที่ฝนตกปลากินเบ็ดดีหรือไม่ ซึ่งเป็นอีกเรื่องหนึ่งที่ต้องพิสูจน์กัน ความเป็นเท็จอยู่ที่ผู้พูดได้ยกสถิติมาอ้าง โดยไม่บ่งว่าเป็นสถิติหากเพื่อนคุณเปลี่ยนคำพูดเสียใหม่เป็น *"ไปตกปลากันเถอะ วันนี้ฝนตกปลา* **อาจ** *กินเบ็ดดี"* ถ้อยคำนี้ทางตรรกถือว่าเป็นจริง เพราะบ่งว่าเป็นสถิติ จึงขึ้นอยู่กับคุณว่าจะเชื่อสถิติของเพื่อนคุณหรือไม่

ตรรกในโปรแกรม

ผมนำกรรมวิธีทางตรรกมาแสดง จะชี้ให้เห็นว่าปัญหามีอยู่ 2 ลักษณะ กรร ทางตรรกในการแก้ปัญหาก็มีอยู่ 2 แบบ กัน เราเขียนโปรแกรมคอมพิวเตอร์ก็เพื่ ปัญหา ดังนั้นกรรมวิธีในการเขียนโปรแ จึงควรมีอยู่ 2 รูปแบบเช่นกัน

แต่เท่าที่ผ่านมา เราเขียนโปรแกร การแก้ปัญหาเกี่ยวกับหนทางปฏิบัติเท่ จึงดูเสมือนว่ามีแบบอย่างการเขียนโปรแ อยู่แบบเดียว จะเห็นได้จากข้อกำหนดที่ใ ใช้ในการเขียนโปรแกรม เช่น การเขียนโ แกรมจากบนลงล่าง(top-down design การเขียนขยายความ(stepwise refinem การเขียนโปรแกรมแบบเป็นโมดูล(modu programming) และการเขียนโปรแกรมแ เป็นโครงสร้าง(structure programmi ล้วนแล้วแต่เป็นการตีกรอบให้เราเขียนโ แกรมอย่างเป็นขั้นตอนนั่นเอง

ถึงยุคปัจจุบัน เพื่อจะสอนให้คอมพิ เตอร์ฉลาดมากขึ้น เราต้องให้คอมพิวเ แก้ปัญหาเกี่ยวกับข้อเท็จจริงด้วย ทั้งนี้เพ ว่า(ขอให้ดูในรูปที่ 1) ในยุคแรกๆ นั้น

โครงสร้างทางความคิด

ขั้นคิด
1. ระบุอุบัติการณ์
2. ทบทวนตนเอง
3. กำหนดความมุ่งหมาย
4. ระบุตัวปัญหา

ขั้นตัดสินใจ
5. กำหนดหนทางปฏิบัติ
6. เลือกหนทางปฏิบัติ

ขั้นปฏิบัติ
7. วางแผน
8. ดำเนินการ

รูปที่ 1 โครงสร้างทางความคิด หรือ การใช้กรรมวิธีทางตรรกในกา ปัญหา เกี่ยวกับการเลือกหน ปฏิบัติ

ขียนโปรแกรมให้คอมพิวเตอร์ทำงานในขั้น
ฏิบัติ ต่อมาเราให้เลือกหนทางปฏิบัติให้แก่
าด้วย ขณะนี้และต่อไปเราจะให้มันกำหนด
นทางปฏิบัติให้แก่เราด้วยว่า ในการแก้
ญหานี้ จะมีหนทางปฏิบัติอยู่กี่ทาง เช่น
ามตัวอย่างที่ยกมา คอมพิวเตอร์อาจกำหนด
นทางปฏิบัติตามที่ผมกล่าวและอาจมีมาก
ว่า เช่น ลงทุนเล่นแชร์ ซึ่งทุกหนทางปฏิบัติ
ะไปเกี่ยวข้องกับข้อเท็จจริง

(ตามที่ผมกล่าวว่าปัญหาแยกออก
น 2 ลักษณะนั้น เพราะแยกมันออกมา
ต่ความเป็นจริงแล้วมักจะปนกันอยู่)

เมื่อเป็นเช่นนี้ แบบอย่างการเขียน
รแกรมจึงจะต้องเพิ่มขึ้นมาอีกหนึ่ง แต่
าคงนำกรรมวิธีทางตรรกมาใช้โดยตรง
ได้ เช่นเดียวกับที่เราไม่สามารถ นำกรรม
ธีทางตรรกมาใช้โดยตรง ในการแก้ปัญหา
ยวกับหนทางปฏิบัติ (จึงมีข้อกำหนดดังที่
ว่ามา) สำหรับในการแก้ปัญหาข้อเท็จจริง
ไม่มีข้อกำหนดเป็นกรรมวิธีที่เด่นชัด คงมี
บ้างในตัวภาษาที่จะนำมาใช้ เช่น ในภาษา
รล็อก จึงต้องรอเวลาอีกสักระยะ ขณะนี้
คงต้องเขียนโปรแกรมตามรูปแบบเท่าที่มี
ะตามถนัดไปก่อน (เทียบได้กับที่เคยเขียน
มาในภาษาเบสิก)

เรื่องที่กล่าวมานี้ เป็นเรื่องที่ต้องทำ
ามเข้าใจในขั้นหนึ่งก่อน และโปรดทราบ
ยว่า ภาษาโปรล็อกเอื้ออำนวยให้เราเขียน
รแกรมทั้งในการแก้ปัญหาข้อเท็จจริง และ
ญหาเกี่ยวกับหนทางปฏิบัติ แบบอย่างการ
ียนโปรแกรมจึงมีทั้ง 2 แบบ และทั้งแยก
ได้อย่างเด่นชัดหรือนำมารวมกัน แต่ในที่
ก็ต้องนำมารวมกันตามธรรมชาติของการ
ปัญหา

เพราะมันปนกันอยู่เช่นนี้ แต่ในบาง
งจำเป็นต้องแยกออกเพื่อชี้ให้เห็นเด่นชัด
จะขอเรียกกรรมวิธีในการเขียนโปรแกรม
บดั้งเดิมที่ใช้กันมาว่าเป็น **แบบขั้นตอน**
นแบบใหม่นี้ว่า **แบบสุ่ม(random)** ซึ่งผม
ว่าน่าจะช่วยในการทำความเข้าใจกับภาษา
โปรล็อกได้ดีขึ้น เพราะหนังสือเกี่ยวกับภาษา
โปรล็อกมักจะขึ้นต้นด้วยแบบสุ่ม ตามด้วย
แบบขั้นตอน ก่อให้เกิดความฉงนแก่ผู้เริ่ม
ศึกษาว่าจะเอาอย่างไรกันแน่ และผมขอเริ่ม
เรื่องของโปรล็อกเสียที

ดูจากภายนอก

เนื่องจากตัวแปรภาษาโปรล็อกเป็น
อินเตอร์พรีตเตอร์ ในด้านการใช้งานจึงคล้าย
กับภาษาเบสิกมากโดยเฉพาะบนเครื่องไอบีเอ็ม
หลังจากการบูทดอสแล้ว เรารันตัวแปรภาษา
เบสิกโดยคีย์คำว่า BASIC สำหรับภาษา
โปรล็อกก็เช่นกันทำโดยคีย์คำว่า PROLOG

เมื่ออยู่ในภาษาเบสิกแล้ว คุณสามารถ
ใช้คำสั่งตรง เช่น LOAD, SAVE และอื่นๆ
ในภาษาโปรล็อกก็ทำได้เช่นกัน แต่โปรล็อก
เรียกวิธีการนี้ว่า เป็นการเรียกใช้โปรแกรม
ย่อย

ในขั้นป้อนโปรแกรมก็ดำเนินการเช่น
เดียวกันอีก แตกต่างกันตรงที่ภาษาเบสิกใช้
หมายเลขบรรทัด และโดยปกติจะต้องป้อน
โปรแกรมตามลำดับแต่ต้นจนจบ ส่วนใน
ภาษาโปรล็อกไม่มีหมายเลขบรรทัด การป้อน
โปรแกรมไม่มีลำดับก่อนหลัง และไม่มีคำว่า
จบโปรแกรม เพราะจะป้อนจากข้อมูลที่มีและ
ที่ได้มาใหม่ไปเรื่อยๆ (จนเต็มหน่วยความจำ)

ในขั้นการรันตรงนี้ครับที่แตกต่างกัน
ก่อนอื่นขอทำความเข้าใจก่อนว่า โปรล็อก
เป็นภาษาแบบโต้ตอบ คือผู้ใช้ต้องอยู่หน้า
เครื่อง(หรือเทอร์มินัล) เพื่อถามตอบกับ
เครื่อง การรันโปรแกรม(หากและซึ่งยังคงใช้
คำนี้อยู่) ในภาษาโปรล็อกทำได้ 2 แบบคือ
โดยการเรียกใช้โปรแกรมย่อย(ของระบบ
ดังกล่าวมาแล้ว หรือที่สร้างขึ้นใหม่ดังจะ
กล่าวต่อไป) อีกวิธีหนึ่งโดยการตั้งคำถาม
ถามเครื่องในเรื่องที่ป้อนไปแล้ว

การรันโปรแกรมทั้งสองแบบนี้ ที่ผม
เรียกว่าเป็นแบบสุ่ม เพราะเราจะป้อนข้อมูล
ในเรื่องใดกี่เรื่องก็ได้ จะป้อนเรื่องใดก่อนหลัง
ก็ได้ ในการถามเราจะเลือกถามเรื่องใดก่อน
หลังก็ได้ จะถามเพียงบางข้อหรือทั้งหมดก็
ได้ เช่น ป้อนข้อมูลเกี่ยวกับลักษณะอากาศ
เกี่ยวกับตลาดหลักทรัพย์ และเกี่ยวกับการ
สำรวจสำมะโนประชากรรวมไว้ด้วยกัน(แต่
โดยปกติจะแยกเรื่องกัน) ในการถามเราอาจ
ถามเกี่ยวกับตลาดหลักทรัพย์เพียงเรื่องเดียว
และเพียงข้อเดียวก็ได้ นอกจากนี้เรายังสามารถ
ป้อนข้อมูลไป ถามไปก็ได้เช่นกัน ส่วนโปร-
แกรมย่อยก็เช่นกัน เราจะป้อนไปที่โปร-
แกรมก็ได้ และจะเลือกรันโปรแกรมใดก็ได้

ที่เรียกว่าโปรแกรม

รูปแบบของภาษาโปรล็อกแตกต่างกับ
ภาษาตามแบบมาก แม้แต่คำว่า โปรแกรม
ก็ไม่ควรนำมาใช้ในภาษานี้ เพราะโปรแกรม
ในภาษาโปรล็อกแบ่งออกเป็น 3 ส่วน คือ
ข้อเท็จจริง กติกา และคำถาม(ซึ่งแต่ละเรื่อง
นี้คืออะไรผมจะกล่าวต่อไป) ที่แปลกคือ
คำถามอยู่ในขั้นตอนของการรันโปรแกรม
แต่ก็นับว่าเป็นส่วนหนึ่งของการโปรแกรม
ด้วย เพราะในการเขียนโปรแกรม จะต้อง
กำหนดว่าจะให้ถามอะไรและอย่างไร

ในหัวข้อที่แล้วผมใช้คำว่าโปรแกรม
กับข้อมูลปนกัน ที่เป็นเช่นนี้เพราะ ที่ว่าป้อน
โปรแกรมหรือป้อนข้อมูลนั้น สิ่งที่ป้อนจริงๆ
คือ ข้อเท็จจริงกับกติกา และมีเพียง 2 สิ่งนี้
เท่านั้น แต่เมื่อป้อนแล้วจะรวมเรียกว่าฐาน
ข้อมูล(Data base) ดังนั้นจึงอาจเรียกว่า เป็น
โปรแกรมหรือข้อมูลก็ได้ ตามแต่เราจะดูให้
เป็นอะไร แต่ตามความรู้สึกของผู้เขียนโปร-
แกรมแล้ว จะเห็นว่ากติกาคือโปรแกรมที่
แท้จริง และโปรแกรมย่อย(procedure) ที่ผม
เคยเอ่ยถึงก็คือรูปแบบหนึ่งของกติกา

อีกเรื่องหนึ่งที่ผมควรกล่าวถึงก่อน
คือ เพียงข้อเท็จจริงกับคำถาม หรือกติกาใน
ฐานะโปรแกรมย่อยและการเรียกใช้ ต่างก็เป็น
โปรแกรมในภาษาโปรล็อกโดยสมบูรณ์ แต่
ที่สมบูรณ์จริงๆ ควรประกอบด้วย 3 ส่วนนี้
(โดยถือว่าคำถามกับการเรียกใช้โปรแกรม
ย่อยเป็นเรื่องเดียวกัน)

การดูรูปแบบของภาษาโปรล็อกให้เห็น
เด่นชัด จึงต้องดูที่ 3 เรื่องนี้ และผมขอย้ำ

อีกครั้งว่า ในภาษาโปรล็อกมีเรื่องที่จะศึกษาอยู่ 3 เรื่องนี้เท่านั้น ซึ่งจะแตกเป็นเรื่องปลีกย่อยมากพอสมควร แต่ในฉบับนี้ผมจะนำมากล่าวโดยย่อ เพื่อให้เห็นรูปแบบของภาษาโปรล็อกได้ง่าย

ข้อเท็จจริง (facts)

คำว่า "ข้อเท็จจริง" ในภาษาไทยเราบ่งความหมายอย่างชัดเจน ผมไม่ทราบว่าชนชาติที่พูดภาษาอังกฤษจะมีความรู้สึกต่อคำว่า "Facts" เหมือนกับเราในคำไทยหรือไม่ เพราะข้อเท็จจริงไม่ใช่ความจริง(ซึ่งต้องจริงตลอดกาล) เช่นกล่าวว่า นาย a เป็นพ่อนาย b เป็นข้อเท็จจริง เพราะจะเป็นจริงต่อเมื่อนาย b เกิดแล้ว (อย่าซีเรียสครับ : ชูโม่)

ที่กล่าวมานี้เป็นชีวิตจริง แต่ในภาษาโปรล็อกถือว่า ข้อเท็จจริงจะเป็นจริงต่อเมื่อเราได้ป้อนข้อมูลคือข้อเท็จจริงนี้ให้แก่มันแล้ว และยังคงอยู่ในหน่วยความจำ (ตัวมันเองไม่มีปัญญาไปสืบหรอกครับว่า นาย a เป็นพ่อนาย b หรือไม่ มันเชื่อตามที่เราป้อน) เรื่องนี้ดูเป็นเรื่องเล็กแต่สำคัญมาก เพราะเป็นบันไดขั้นแรก ในการทำความเข้าใจกับภาษาโปรล็อก ผมจึงขอกล่าวย้ำอีกครั้งว่า ข้อเท็จจริงจะเป็นจริงในภาษาโปรล็อก ต่อเมื่อมีข้อเท็จจริงนั้นอยู่ในฐานข้อมูลคือในหน่วยความจำ

ข้อเท็จจริงเรื่องพ่อตามที่กล่าวมา จะเขียนเป็นภาษาโปรล็อกตามแบบที่ใช้กันอยู่ทั่วๆ ไปได้ว่า

father (a,b).

ขอให้ดูในรูปที่ 2 ซึ่งผมได้ป้อนข้อเท็จจริงไว้จำนวนหนึ่ง ขอให้สังเกตด้วยว่าผมปิดท้ายด้วยจุด แสดงถึงการจบอนุประโยค (clause) คำแต่ละคำเช่น father, a และ b ภาษาโปรล็อกถือว่าเป็นค่าคงตัว(constant) และต้องเขียนขึ้นต้นด้วยอักษรเล็ก ส่วนจะยาวได้กี่อักขระนั้น ผมยังไม่พบว่ากำหนดไว้ ณ ที่ใด และสามารถใช้ขีดล่างเพื่อช่วยให้อ่านได้ง่าย ที่ผมตั้งชื่อคนไว้เพียงอักขระเดียวในตัวอย่าง เพื่อให้เห็นเด่นชัดเท่านั้นครับ

```
: father(a,b).
: father(b,c).
: father(x,y).
: father(y,z).
: father(x,q).
```

รูปที่ 2 ข้อเท็จจริงเกี่ยวกับ father ที่ป้อนเข้าเครื่อง

คำถาม (Question)

จากนี้เราก็สามารถตั้งคำถามกับเครื่องได้แล้ว ซึ่งถามได้หลายแบบ แต่อาจแบ่งได้เป็น 3 ลักษณะใหญ่ๆ ขอนำมาแสดงเพียง 2 ลักษณะก่อน คือ ถามว่า

นาย a เป็นพ่อนาย b ใช่ไหม?

กับ ใครเป็นพ่อนาย b?

หรือ นาย a เป็นพ่อใคร?

ได้แสดงการตั้งคำถามพร้อมด้วยคำตอบในรูปที่ 3 ขอให้สังเกตด้วยว่าผมปิดท้ายด้วย ? แสดงว่าเป็นประโยคคำถาม คำถามแรกคือ father(a, b)? เป็นการถามว่านาย a เป็นพ่อนาย b ใช่ไหม? ซึ่งเครื่องตอบว่า yes หมายถึงใช่ คือเราได้ให้ข้อเท็จจริงข้อนี้ไว้แล้ว คำถามที่ 2 เป็นการถามว่านาย e เป็นพ่อนาย f ใช่ไหม? เครื่องตอบว่า no คราวนี้ไม่ได้หมายความว่าไม่ใช่ หากหมายความว่าไม่ทราบ เพราะเราไม่ได้ให้ข้อเท็จจริงไว้

```
: father(a,b).
: father(b,c).
: father(x,y).
: father(y,z).
: father(x,q).
:
: father(a,b)?
** yes
: father(e,f)?
** no
:
: father(Who,b)?

Who = a
:
: father(a,X)?

X = b
```

รูปที่ 3 แสดงคำถามพร้อมกับคำตอบ

คำถามที่ 3 เป็นการถามว่าใครเป็นพ่อนาย b? ผมใช้คำว่า Who เพื่อให้ใกล้เคียงกับความเป็นจริง แต่ความจริงแล้ว Who เป็นตัวแปรซึ่งเราจะใช้คำว่าอะไรก็ได้ แต่ต้องขึ้นต้นด้วยอักษรใหญ่จึงจะเป็นตัวแปร เครื่องตอบว่า Who = a ซึ่งเราจะอ่านเป็นว่าใคร คนนั้นคือนาย a

คำถามที่ 4 เป็นการถามว่านาย a เป็นพ่อใคร? คราวนี้ผมใช้ตัวแปรง่ายๆ ว่า X และจะเห็นว่า X ตัวนี้เป็นตัวแปร และเป็นคนละตัวกับ x ซึ่งเป็นค่าคงตัว เครื่องตอบว่า X = b ซึ่งเราต้องอ่านช่วยอีกเช่นกันว่าใคร คนนั้นคือนาย b

จากที่กล่าวมานี้ จะเห็นว่าผมตั้งคำถามเกี่ยวกับครอบครัวของตระกูล a เท่านั้น และเฉพาะ a กับ b โดยไม่เกี่ยวข้องกับ c และตระกูล x เลย และการตั้งคำถามเพียงเท่านี้ก็ถือว่าเป็นการเขียน ป้อน และรันโปรแกรมในภาษาโปรล็อกโดยสมบูรณ์แล้ว จึงได้กล่าวว่าลักษณะของโปรแกรมในภาษาโปรล็อกเป็นแบบสุ่ม

กติกา (Rules)

ความจริงผมควรใช้คำว่า "กฎ" แทนคำในภาษาอังกฤษว่า "Rules" แต่ผมเห็นว่าใช้คำว่า "กติกา" น่าจะเหมาะสมกว่า การใช้ภาษาโปรล็อกจะไม่กว้างขวาง ถ้าไม่มีรูปแบบของกติกาอยู่ในภาษาโปรล็อก และกติกานี้เองที่ทำให้เราสามารถเขียนโปรแกรมอย่างเป็นขั้นตอนได้ ขอแสดงตัวอย่างให้ดูก่อน จากที่เราให้ข้อเท็จจริงในเรื่องพ่อ เราสามารถนำมาตั้งเป็นกติกาในเรื่องปู่ได้ เช่น

นาย X เป็นปู่นาย Z ถ้า

นาย X เป็นพ่อนาย Y และ

นาย Y เป็นพ่อนาย Z

ซึ่งเขียนเป็นภาษาโปรล็อกและป้อนเข้าเครื่องดังรูปที่ 4 ขอให้สังเกตด้วยว่า X และ Z เป็นตัวแปร จากนี้เราก็สามารถ

```
: father(a,b).
: father(b,c).
: father(x,y).
: father(y,z).
: father(x,q).
:
: grandpa(X,Z) :-
:    father(X,Y),
:    father(Y,Z).
:
: grandpa(a,Who)?

Who = c
:
: grandpa(Who,z)?

Who = x
:
: father(x,Who)?

Who = y

Who = q
```

ปที่ 4 แสดงถึงการตั้งกติกาในเรื่อง grandpa พร้อมด้วยคำถามและคำตอบ

ด้ว่าใครเป็นปู่ใคร หรือปู่ของใครคือใคร
งแสดงในรูปเดียวกันนี้ โดยคำถามแรกถาม
า นาย a เป็นปู่ใคร? เครื่องตอบว่านาย c
ะเห็นว่าตรงกับข้อเท็จจริงที่เราให้ไว้ว่า
าย a เป็นพ่อนาย b กับนาย b เป็นพ่อนาย
คำถามต่อไปถามว่า ปู่ของนาย z คือใคร?
รื่องตอบว่านาย x ผมถามต่อไปว่า นาย x
นพ่อใคร? เครื่องตอบว่านาย y กับนาย
เพราะในข้อเท็จจริงผมให้ไว้ว่านาย x มีลูก
คน

จะเห็นว่าหากเราป้อนข้อมูลจากการ
รวจสำมะโนประชากรเป็นข้อเท็จจริง และ
ยนกติกาที่เหมาะสม เราจะสามารถสืบสาว
วเรื่อง เรื่องของครอบครัวใดครอบครัว
นึ่ง ซึ่งจะไปสัมพันธ์กับอีกหลายๆ ครอบครัว
ย่างง่ายดาย ซึ่งจะเขียนได้ยากมากใน
าษาตามแบบ

ปรแกรมย่อย (Procedure)

ตามที่กล่าวแล้วว่าการรันโปรแกรมใน
าษาโปรล็อก ทำด้วยการตั้งคำถามวิธีหนึ่ง
วิธีหนึ่งโดยการเรียกใช้โปรแกรมย่อยโดย
ง ซึ่งอาจเป็นการเรียกใช้โปรแกรมย่อย
ตรฐาน คือมีอยู่แล้วในโปรล็อก ซึ่งต่อไป

```
%  The Tower of Hanoi Game

hanoi(N) :- move(N,left,center,right).

move(0,_,_,_) :- !.
move(N,A,B,C) :-
          M is N - 1,
          move(M,A,C,B),
          inform(A,B),
          move(M,C,B,A).

inform(X,Y) :-
          print('Move the disk on ',X,' to ',Y),
          nl.
```

รูปที่ 5 โปรแกรมย่อย หอคอยแห่งเมืองฮานอย เครื่องหมาย % หมายถึง ข้อความในบรรทัดนี้เป็นหมายเหตุซึ่งเครื่องจะไม่นำไปใช้

ผมจะขอเรียกว่าคำเดิม หรือที่ผู้เขียนโปรแกรมจะสร้างขึ้นใหม่ โปรแกรมย่อยที่เขียนขึ้นใหม่นี้ คือรูปแบบหนึ่งของกติกา ซึ่งต่อไปผมจะขอเรียกว่าคำใหม่

ขอให้ดูตัวอย่างในรูปที่ 5 ซึ่งเป็นโปรแกรมย่อยแก้ปัญหา "หอคอยแห่งเมืองฮานอย" เป็นเกมปัญหาลับสมอง โดยกำหนดว่ามีจานอยู่จำนวนหนึ่ง มีขนาดไม่เท่ากัน ตรงกลางเจาะรู วางซ้อนอยู่บนหลักอันหนึ่ง เรียงลำดับจากใหญ่ไปหาเล็ก โดยให้อันเล็กสุดอยู่บน และมีหลักอีก 2 อัน ให้ผู้เล่นย้ายจานทั้งหมดไปอีกหลักหนึ่ง โดยมีกติกาว่าให้ย้ายได้ทีละ 1 จาน โดยที่ทุกขณะจานใบใหญ่จะวางซ้อนบนใบเล็กไม่ได้ ใช้หลักที่เหลืออีก 1 หลักเป็นที่พักจานได้ (หากต้องการทราบรายละเอียดของการแก้ปัญหา ขอให้ดูในไมโครฯ ฉบับที่ 2 และ 4)

ที่ผมนำตัวอย่างนี้มาแสดงก็เพื่อที่นอกจากจะแสดงว่าเราสร้างโปรแกรมย่อยในภาษาโปรล็อกได้อย่างไรแล้ว ยังแสดงว่าเราสามารถใช้กรรมวิธีเรียกตนเอง(recursive) ในภาษาโปรล็อกได้ ซึ่งไม่มีในอีกหลายภาษา เช่น FORTRAN และ BASIC (และความจริงแล้วโปรล็อกใช้กรรมวิธีของการเรียกตนเองอย่างเป็นของธรรมดา (ในภาษาอื่นที่เรียกตนเองได้เช่น ปาสคาลหรือซี จะใช้กรรมวิธีนี้เมื่อจำเป็นเท่านั้น)

ตัวอย่างนี้เป็นการสร้างโปรแกรมย่อยให้ชื่อว่า hanoi โดยให้ hanoi เรียกใช้ move และใน move มีการเรียกตนเอง ขอให้สังเกตวิธีการเขียนโปรแกรมย่อย move ซึ่งเป็นแบบอย่างของการเรียกตนเอง อย่าพึ่งสนใจหรือทำความเข้าใจในเรื่องของโปรแกรม หรือรายละเอียดในตอนนี้ครับ เพราะยังมีเรื่องอีกมากที่ผมยังไม่พูดถึง ตอนนี้ขอให้ดูรูปแบบของโปรแกรมย่อยเท่านั้นก่อน อีกโปรแกรมหนึ่งที่ move เรียกใช้คือ inform ใน inform มีการเรียกใช้ print และ nl(newline) ซึ่งเป็นคำเดิม

จะเห็นว่าการสร้างโปรแกรมย่อย move ใช้กรรมวิธีของการเขียนโปรแกรมอย่างมีขั้นตอน มีการสร้างโปรแกรมย่อยและเรียกใช้โปรแกรมย่อยต่อเนื่อง ในแบบของการสร้างคำใหม่ ในรูปแบบที่คล้ายกับโปรแกรมในภาษาซีมาก

การป้อนโปรแกรมนี้ ขอให้คุณป้อนต่อจากโปรแกรมที่ป้อนไว้เดิมได้ (ดังที่กล่าวแล้วว่า โปรแกรมในภาษาโปรล็อกเป็นแบบสุ่ม) สำหรับการรันให้พิมพ์คำว่า hanoi ต่อด้วยจำนวนจานที่ต้องการ เช่น

```
hanoi (4)!
```

ขอให้สังเกตด้วยว่าผมปิดท้ายด้วย ! แสดงว่าเป็นประโยคคำสั่ง แต่เครื่องหมายนี้มีความหมายเป็น 2 นัย ที่เห็นอยู่ใน move หมายถึง cut คือให้หยุด ซึ่งจะกล่าวต่อไป

เมื่อรันโปรแกรมนี้แล้ว เครื่องจะบอกให้เราย้ายจานจากหลักใดไปหลักใด ซึ่งเมื่อคุณทำตามลำดับที่เครื่องบอก ก็จะแก้ปัญหา

หอคอยแห่งเมืองฮานอยได้ ที่ผมให้ไว้เพียง 4 จาน เพื่อให้เห็นบนจอภาพได้หมด หากใส่จานมากกว่านี้จะไม่เห็นตอนต้นๆ ผลการรันดังรูปที่ 6

คุณที่เคยชินกับภาษาเบสิกอาจงง เพราะในภาษาเบสิกไม่มีกรรมวิธีในการสร้างคำใหม่ จึงไม่มีวิธีการรันโปรแกรมแบบนี้ คุณที่เคยชินกับภาษาปาสคาลหรือซีที่มีการสร้างคำใหม่ก็มีสิทธิงงได้เช่นกัน เพราะตัวแปรภาษาของทั้งสองนี้เป็นแบบคอมพายเลอร์ ไม่มีสิทธิเรียกใช้โปรแกรมย่อยโดยตรง ต้องเขียนโปรแกรมนี้เป็นเอกเทศ และรันจากชื่อไฟล์ในดิสค์ที่โปรล็อกทำได้เพราะในโปรล็อกสร้างคำใหม่ได้ และตัวแปรภาษาเป็นอินเทอร์พริตเตอร์

เครื่องหมาย

เรื่องของเครื่องหมาย ความจริงแล้วเป็นส่วนหนึ่งของไวยกรณ์(syntax) ของภาษา ซึ่งตั้งใจจะนำไปกล่าวในฉบับต่อไป แต่ผมเห็นว่ามีความจำเป็นที่จะต้องนำมากล่าวในตอนนี้ก่อน คือ

- . จุดใช้สำหรับปิด(อนุ)ประโยค (clause) ที่เป็นข้อเท็จจริง และกติกา
- ? ใช้สำหรับปิดประโยคคำถาม
- ! มีความหมายเป็น 2 นัยคือ ใช้สำหรับปิดประโยคที่เป็นคำสั่ง ดังที่ใช้มาแล้ว อีกนัยหนึ่งคือ cut ดังที่จะกล่าวต่อไป
- :- หมายถึง IF คือ ถ้า แต่ในบางครั้งเราจะมีความรู้สึกเหมือนให้กระทำดังจะกล่าวต่อไป
- , หมายถึง AND ตามคณิตศาสตร์บูลีน
- ; หมายถึง OR ตามคณิตศาสตร์บูลีน

เงื่อนไข

ภาษาตามแบบโดยเฉพาะที่เป็นโครงสร้าง จะมีแบบของเงื่อนไขให้เราเลือกใช้มากมาย เช่น IF, CASE, FOR, WHILE และ UNTIL แม้แต่ในภาษาเบสิกอย่างน้อยก็มี IF และ FOR แบบของเงื่อนไขดังกล่าวมานี้ที่สำคัญคือ IF และ WHILE ซึ่งมีในภาษาโปรล็อกเช่นกัน แต่ถ้าดูโดยผิวเผินจะเห็นว่ามีแต่ IF เท่านั้น ทั้งนี้เพราะ WHILE ไปแอบแฝงอยู่ในเรื่องการเรียกตนเอง

ผมจึงจะนำเรื่องของ IF มากล่าวถึงก่อนในหัวข้อนี้ และแม้แต่เรื่อง IF เองก็ดูได้ยากสำหรับผู้ที่คุ้นเคยกับการเขียนโปรแกรมตามแบบ เช่นที่ผมเขียนเป็นกติกาไว้ว่า

นาย X เป็นปู่ของนาย Z IF
นาย X เป็นพ่อของ Y AND
นาย Y เป็นพ่อของนาย Z.

หากเขียนในรูปของโปรแกรมตามแบบจะเขียนได้เป็น

IF (นาย X เป็นพ่อของนาย Y) AND (นาย Y เป็นพ่อของนาย Z) THEN นาย X เป็นปู่ของนาย Z;

จะเห็นว่าโปรแกรมตามแบบ ระบุเงื่อนไขก่อน แล้วจึงให้ดำเนินการหากเงื่อนไขเป็นจริง หรืออาจกล่าวอีกนัยหนึ่งว่า หากเงื่อนไขเป็นจริงจะได้ว่าข้อยุติเป็นจริง แต่ในภาโปรล็อกกลับสลับกันคือ อ้างถึงข้อยุติก่อคือ ข้อยุติจะเป็นจริงถ้าเงื่อนไขเป็นจริง

ผู้ที่คุ้นเคยกับการเขียนโปรแกรมตแบบอาจงง จนเขียนกติกาในภาษาโปรล็ไม่ถนัด หรือไม่ก็พยายามคิดตามแบบเก่ก่อนแล้วสลับกลับ ซึ่งผมเห็นว่า ถ้าเราเขียนโปรแกรมในภาษาใดควรคิดอย่างภานั้นโดยตรง อย่างในภาษาโปรล็อกนี้ ก็คจะนึกถึงข้อยุติก่อนแล้วจึงนึกถึงเงื่อนไข ซึ่เป็นโดยธรรมชาติอยู่แล้ว ตัวเราเองต่างหาถูกฝืนความรู้สึก เมื่อเขียนโปรแกรมตามแจนติดนิสัย เรื่องนี้เป็นอีกเรื่องหนึ่งที่ต้องความเข้าใจ และปรับความรู้สึกให้ได้ จึงกเป็นเรื่องโชคดีของผู้ที่ไม่เคยศึกษาโปรแกในภาษาใดมาก่อน

คำถามกับคำสั่ง

คำสั่งในที่นี้ของผมหมายถึง การเรีใช้โปรแกรมย่อยโดยตรง จากที่กล่าวมาแว่า โปรล็อกถือว่าข้อยุติจะเป็นจริงต่อเมื่อเง

```
: hanoi(4)!
Move the disk on left to right
Move the disk on left to center
Move the disk on right to center
Move the disk on left to right
Move the disk on center to left
Move the disk on center to right
Move the disk on left to right
Move the disk on left to center
Move the disk on right to center
Move the disk on right to left
Move the disk on center to left
Move the disk on right to center
Move the disk on left to right
Move the disk on left to center
Move the disk on right to center
```

รูปที่ 6
ผลการรันโปรแกรม
หอคอยแห่งเมืองฮาน

ขเป็นจริง หรือทางซ้ายจะเป็นจริงต่อเมื่อทาง
วาเป็นจริง หรือทางขวาทั้งหมดเป็นจริง
เมื่อมีหลายอนุประโยคและเราต่ออนุประโยค
หล่านี้ด้วย AND คือ ,) ดังนั้นโปรล็อกจะ
รวจประโยคทางขวาทุกประโยคว่าเป็นจริง
หรือไม่ หากพบประโยคที่ให้มีการปฏิบัติ
ช่น ให้อินพุทหรือเอาท์พุทก็จะปฏิบัติตาม
ระโยคนั้นๆ ด้วย

ช่น

```
llo :-
    print ('This is my first program in prolog'),
    print (' 5 Apr 2529'),
    father (q,r).
```

เท่ากับเป็นการสร้างโปรแกรมย่อย
llo ซึ่งไม่มีค่าใดๆ เลย การอ่านโปรแกรม
จะต้องตีความหมายว่า hello เป็นจริง ถ้า
ที่พิมพ์ข้อความ 2 บรรทัดนั้น และนาย q
นพ่อนาย r

ต่อไปหากเราตั้งคำถามว่า hello? เครื่อง
พิมพ์ข้อความทั้ง 2 บรรทัดนั้นและตอบว่า
เพราะเรายังไม่เคยให้ข้อเท็จจริงไว้ว่านาย
เป็นพ่อนาย r (และถ้าเราป้อนคำว่า print
เป็น prit โปรล็อกจะให้เป็นเท็จทันที
าะไม่มีคำนี้อยู่ในฐานข้อมูล และออกจาก
รแกรมนี้ให้คำตอบว่า no)

หากเราใช้คำสั่งว่า hello! เครื่องจะ
พ์ข้อความทั้ง 2 บรรทัดนี้อีกเช่นกัน แต่
ตอบว่า yes หรือ no ทั้งนี้เพราะโปรล็อก
จะเพียงแต่ตรวจว่าทางขวาเป็นจริงหรือไม่
แต่ไม่แสดงผล คำถามกับคำสั่งจึงแตกต่างกัน
เพียงเท่านี้ และขึ้นกับเราว่าจะให้เป็นคำถาม
หรือคำสั่ง (แต่ถ้าเขียนโปรแกรมผิด คำถาม
กับคำสั่งจะให้ผลต่างกัน)

โปรแกรมย่อยเรียกตนเอง (recursive)

ตามที่กล่าวแล้วว่า โปรล็อกใช้การเรียกตนเองแทนเงื่อนไข WHILE ผมจึงขอแสดงกรรมวิธีของการเรียกตนเองก่อน หลักการของการเรียกตนเองมีข้อกำหนดอยู่ 2 ข้อ คือ การเรียกตนเองต้องมีจุดจบที่แน่นอน กับต้องมีอยู่หนึ่งครั้งที่ไม่เรียกตนเอง ในการเขียนโปรแกรมจริงข้อกำหนดทั้ง 2 ข้อนี้ อาจแยกกันหรือรวมกันก็ได้ แต่ส่วนมากมักรวมกันโดยเฉพาะในโปรล็อก

ขอให้ดูในรูปที่ 7 ตอนบนซึ่งเป็นส่วนของโปรแกรม โปรแกรมนี้จะพิมพ์ข้อความว่า How do you do? และค่าของ N เป็นจำนวนครั้งเท่ากับที่เราผ่านค่าไปให้ ดังแสดงในตอนล่างของรูป

การอ่านโปรแกรมนี้ ขอให้อ่านดังที่ผมกล่าวมาแล้ว เช่น ในประโยคแรกขอให้อ่านว่า greeting(0) เป็นจริงเมื่อ cut คือ ! เป็นจริง ซึ่งโปรล็อกจะต้องกระทำ cut เพื่อตรวจสอบว่าเป็นจริงหรือไม่ การ cut หมายถึงไม่ต้องตรวจหาคำว่า greeting อีกต่อไป เท่ากับเป็นการยกเลิกการเรียกตนเอง และตรงตามข้อกำหนดของการเรียกตนเองคือมีจุดจบที่แน่นอนคือเมื่อ N เป็นศูนย์ และไม่มีการเรียกตนเองในประโยคนี้

ในการรันโปรแกรมตามตัวอย่าง เท่ากับเป็นการเรียกใช้ greeting(5) โปรล็อกจะพยายามหา greeting(5) ซึ่งไม่มี จึงได้แก่ greeting ที่เหมาะสมที่สุดคือ greeting(N) จึงให้ค่า 5 แก่ N และเริ่มรันโปรแกรมย่อยนี้ ซึ่งจะพิมพ์ข้อความดังกล่าว นำค่า N มาลบด้วย 1 ให้แก่ M แล้วเรียกใช้ greeting อีก ซึ่งโปรล็อกจะค้นหา greeting ที่เหมาะสมจะได้แก่ greeting(N) อีกเป็นการเรียกตนเองทำซ้ำอยู่เช่นนี้ จนกระทั่ง M มีค่าเป็น 0 เป็นการเรียกใช้ greeting(0) ทำให้ยุติการเรียกตนเองดังกล่าวแล้ว

ข้อสังเกต ตามที่ผมกล่าวว่าโปรล็อกเป็นภาษาแบบสุ่มผสมกับแบบขั้นตอนนั้น ในบางกรณีกลับเป็นแบบเรียงลำดับดังเช่นในโปรแกรมนี้ เราจะป้อน greeting(0) ไว้หลัง greeting(N) ไม่ได้ เพราะโปรล็อกจะค้นหาข้อมูลจากต้นไปปลายฐานข้อมูล และ N เป็นตัวแปรจึงเหมาะกับค่าทุกค่า จึงไม่มีโอกาสไปทำงานใน greeting(0) เลย จะทำซ้ำใน greeting(N) โดยลดค่า M ลงเรื่อยๆ จนติดลบ และจนกระทั่งสแต็คเต็ม ออกจากโปรแกรมนี้ อีกเรื่องหนึ่งที่ผมกล่าวว่า โปร-ล็อกใช้การเรียกตนเองแทนวงรอบ WHILE นั้น ผมได้เขียนโปรแกรมเดียวกันนี้ เป็นโปรแกรมย่อยในภาษาปาสคาล ดังแสดงในรูปที่ 8

```
: listing!

greeting(0) :- !.
greeting(N) :-
        print('How do you do? ', N),
        M is N - 1,
        greeting(M).
:
:
: greeting(5)!
How do you do? 5
How do you do? 4
How do you do? 3
How do you do? 2
How do you do? 1
```

ที่ 7 แบบของโปรแกรมย่อยที่เรียกตนเอง

```
A>type greet.pas
PROCEDURE Greeting(N : Integer);
VAR   M : Integer;
BEGIN
  M := N;
  While M > 0 DO
    BEGIN
    WriteLn('How do you do?  ',M);
    M := M - 1;;
    END
END;
```

รูปที่ 8 โปรแกรมย่อย greeting เมื่อเขียนในภาษาปาสคาล

```
: war(X,Y) :-
:         conflict(X,Y),
:         power_than(X,Y),
:         asserta(inwar(X,Y)).
:
: war(alfa,bravo)?
** no
```

```
: listing!

war(X, Y) :-
        conflict(X, Y),
        power_than(X, Y),
        asserta(inwar(X, Y)).
:
```

รูปที่ 9 แสดงการตั้งกติกา war และคำถามก่อนป้อนข้อเท็จจริง

```
: conflict(alfa,bravo).
: conflict(alfa,charlie).
: power_than(alfa,bravo).
:
: war(X,Y)?

X = alfa
Y = bravo
```

```
: listing!

inwar(alfa, bravo).

power_than(alfa, bravo).

war(X, Y) :-
        conflict(X, Y),
        power_than(X, Y),
        asserta(inwar(X, Y)).

conflict(alfa, bravo).
conflict(alfa, charlie).
```

```
: inwar(X,Y)?

X = alfa
Y = bravo
```

รูปที่ 10 แสดงถึงการป้อนข้อเท็จจริง และการตั้งคำถาม กับคำ
เครื่องได้ เพิ่มข้อเท็จจริง inwar ซึ่งได้ลิสต์มาให้ดู

โปรแกรมที่สร้างโปรแกรม

ในภาพยนตร์และนิยายวิทยาศาสตร์สะท้อนให้เห็นความหวาดระแวงของมนุษย์ที่เกรงว่า เครื่องคอมพิวเตอร์จะฉลาดเกินมนุษย์ จนมนุษย์ไม่สามารถจะควบคุมมันได้ คอมพิวเตอร์จะฉลาดเช่นนี้ได้ก็ต่อเมื่อ มันสามารถเขียนโปรแกรมให้กับตัวมันเองได้ ซึ่งเป็นไปไม่ได้ในภาษาตามแบบ แต่เป็นไปได้ในภาษาโปรล็อก(หมายถึงเขียนโปรแกรมกันอย่างธรรมดา)

แต่ตัวอย่างที่จะนำมาแสดงในฉบับนี้จะเป็นเพียงการเพิ่มข้อเท็จจริงให้กับฐานข้อมูล อีกเรื่องหนึ่งที่ผมเคยกล่าวถึงแต่ไม่ได้ย้ำคือ เราไม่จำเป็นต้องป้อนข้อเท็จจริงก่อนกติกาเสมอไป อาจป้อนกติกาก่อนแล้วป้อนข้อเท็จจริงในภายหลังเมื่อใดก็ได้ ซึ่งจะแสดงในตัวอย่างนี้เช่นกันโดยสมมติว่า เราได้ตั้งกติกาไว้ว่า

"ประเทศ X จะทำสงครามกับประเทศ Y ถ้า ประเทศ X ขัดกับประเทศ Y และประเทศ X มีกำลังอำนาจเหนือประเทศ Y"

เมื่อป้อนกติกานี้เข้าเครื่องแล้ว ก็ให้เจ้าหน้าที่ติดตามและป้อนสถานการณ์ของโลกว่า ใครขัดกับใครบ้าง ใครมีอำนาจเหนือใครบ้าง ไปเรื่อยๆ เมื่อผู้บริหารในระดับสูงต้องการทราบว่าใครจะทำสงครามกับใคร ก็เพียงแต่ตั้งคำถามกับเครื่อง แต่เพื่อแสดงลู่ทางของภาษาโปรล็อกในการสร้างโปรแกรมเอง ผมได้จัดคำถามเป็น 2 ชุดคือ war(X,Y) กับ inwar (X,Y)

ขอให้ดูในรูปที่ 9 ท่อนบนเป็นโปรแกรมที่ป้อน และสมมติว่าเจ้าหน้าที่ยังไม่มีข้อเท็จจริงจะป้อน ผู้บริหารเกิดถามเครื่อง ซึ่งเครื่องจะตอบว่าไม่ทราบ เนื่องจากเราไม่ได้ให้ข้อเท็จจริงไว้เลย และจะไม่ทำคำสั่ง aserta ทั้งนี้ตามที่กล่าวแล้วว่าโปรล็อกพยายามจ
ว่า war(X,Y) เป็นเท็จหรือเป็นจริง เมื่อต
ประโยคแรกคือ conflict(X,Y) ปรากฏว่า
เท็จ เพราะไม่มีประเทศใดขัดกับประเท
เลย และเพราะผมใช้ตรรก AND คือ
การเชื่อมอนุประโยค ดังนั้นเมื่อกรณีหนึ่
เท็จก็จะถือว่า war เป็นเท็จ ไม่ตรวจสอบ
อื่นอีกต่อไป และพิมพ์คำว่า no

ต่อมาเมื่อเจ้าหน้าที่ป้อนข้อมูลที่ไ
มา ดังรูปที่ 10 ตอนบน เมื่อผู้บริหารถาม
จะเห็นว่าคราวนี้ conflict เป็นจริงเ
alfa ขัดกับ brave และ power-than เป็
เพราะ alfa มีกำลังเหนือกว่า bravo ถึ
ประโยคที่ 3 ซึ่งเป็นการปฏิบัติ โปรล็อก
ปฏิบัติตามคำสั่ง asserta เพื่อทดสอบว่า

งหรือไม่ ปรากฏว่าเป็นจริงอีก เครื่องบอกชื่อประเทศคู่สงครามให้ผู้บริหารทราบ

คำสั่ง asserta เป็นคำสั่งให้เพิ่มข้อมูลในวงเล็บคือ inwar(X,Y) ไว้ตอนต้นของฐานข้อมูล (assertz ไว้ตอนท้ายของฐานข้อมูล) แต่เนื่องจาก X และ Y เป็นตัวแปร ตัวโยคที่เพิ่มจริงๆ คือ inwar(alfa, bravo) ดังที่ผมลิสต์ให้ดูในรูป จากนี้เราก็สามารถถามคำถาม inwar ได้ดังแสดงในรูป ในการใช้งานจริงควรให้ผู้บริหารถามคำถาม inwar โปรแกรมนี้ไม่ใช่โปรแกรมใช้งาน เพราะมีข้อบกพร่อง ที่นำมาแสดงก็เพื่อแสดงว่าโปรแกรมนี้เพิ่มข้อเท็จจริงให้แก่ฐานข้อมูลโดยตัวของมันเองได้อย่างไร

รคำนวณ

ภาษาโปรล็อกไม่ใช่ภาษาสำหรับการคำนวณ จึงมีแต่การคำนวณขั้นพื้นฐานคือ บวก ลบ คูณ หาร และยกกำลัง กับเลขจำนวนเต็มเท่านั้น แต่ทั้งนี้ไม่ได้หมายความว่า เราจะใช้ภาษาโปรล็อกในการแก้ปัญหาทางคณิตศาสตร์ เช่น ถอดสมการที่สลับซับซ้อนไม่ได้

```
: listing!

area(usa, 2).
area(india, 3).
area(chaina, 4).
area(brazil, 1).

pop(usa, 203).
pop(india, 540).
pop(chaina, 800).
pop(brazil, 108).
:
:
: den(X,D) :-
:         pop(X,P),
:         area(X,A),
:         D is P/A.
:
: den(india,X)?

X = 180
```

11 แสดงการคำนวณในภาษาโปรล็อก

ตรงกันข้ามกลับทำได้และทำได้เป็นอย่างดี เพียงแต่ไม่สามารถให้ผลลัพธ์จากการคำนวณที่นอกเหนือไปจากการคำนวณขั้นพื้นฐานแก่เราได้เท่านั้น ขนาดของเลขจำนวนเต็มขึ้นกับขนาดของเครื่อง ซึ่งสำหรับเครื่อง 16 บิทจะมีค่าได้ตั้งแต่ −32765 ถึง 32764

จุดประสงค์ของการคำนวณ เพื่อช่วยให้เราแก้ปัญหาข้อเท็จจริงได้ดียิ่งขึ้น เช่น ป้อนข้อเท็จจริงของแต่ละเมืองเป็นพื้นที่และจำนวนประชากร เราก็สามารถเขียนกติกาหาความหนาแน่นของประชากรในแต่ละเมืองได้ ดังแสดงในรูปที่ 11

สำหรับตัวกระทำ(operator) ซึ่งก็คือเครื่องหมายที่ใช้ในการคำนวณมีดังนี้คือ

+ − * / mod ^ is

เรียงตามลำดับเป็น บวก ลบ คูณ หาร หาเศษ ยกกำลัง และการให้ค่า สำหรับการหารเป็นการหารแบบปัดเศษทิ้ง หากต้องการทราบค่าที่เหลือให้ใช้ mod ส่วน is เป็นการให้ค่าเลขจำนวนจริงแก่ตัวแปร เช่น

```
X is 17
Y is X / 3
Z is X mod 3
```

จะเป็นการให้ค่า 17 แก่ X ส่วน Y ได้ค่าเป็น 5 และ Z ได้ค่าเป็น 2 ถ้าใช้เครื่องหมายในการคำนวณหลายตัว จะดำเนินการตามลำดับความสำคัญดังนี้ ^ , *, /, mod, + และ − หากต้องการให้ทำในกรณีใดก่อนให้ใส่วงเล็บ เช่น

```
X is 3 + 4 * 2^3 = 35
X is ((3 + 4) * 2)^3 = 2744
```

save และ load

โปรแกรมที่ป้อนแล้วนี้สามารถเก็บลงดิสก์ได้โดยใช้คำสั่ง save ตามด้วยชื่อไฟล์ หากต่อไปต้องการโหลดลงหน่วยความจำให้ใช้คำสั่ง load เช่น

```
save 'myfirst. pro' !
load 'myfirst. pro' !
```

ซึ่งคำสั่ง save จะเก็บข้อมูลทั้งหมดในดาต้าเบสลงไฟล์ ส่วนคำสั่ง load ตามคู่มือบอกว่า จะยกเลิกข้อมูลทั้งหมดในดาต้าเบสเดิม และโหลดของใหม่มาแทนที่ แต่จากการใช้งานปรากฏว่า จะโหลดมาทับกัน คือมาแทนที่เฉพาะข้อมูลเดียวกัน ส่วนข้อมูลเดิมซึ่งไม่ตรงกับข้อมูลใหม่จะยังคงมีอยู่ตามเดิม และเพิ่มข้อมูลใหม่ที่ไม่ซ้ำกับข้อมูลเดิม

ก่อนจบตอน

เท่าที่กล่าวมานี้เป็นการแสดงรูปแบบของภาษาโปรล็อก และการใช้งานอย่างกว้างๆ และอย่างง่ายๆ คุณอาจจะยังไม่กระจ่างนักก็ไม่เป็นไรครับ ในตอนนี้ขอให้เพียงเข้าใจว่า โดยสรุปแล้วโปรแกรมภาษาโปรล็อกแบ่งออกเป็น 3 ส่วนคือ ข้อเท็จจริง กติกา และคำถาม กับการทำงานของโปรแกรมในภาษาโปรล็อกส่วนใหญ่จะเป็นแบบสุ่ม ก็เพียงพอครับ

สำหรับข้อเท็จจริงจะเกี่ยวพันกับแบบของข้อมูล ซึ่งผมยังไม่เคยกล่าวถึง แต่ได้นำมาใช้บ้างแล้ว คือ ข้อมูลที่เป็นสตริงกับเลขจำนวนเต็มที่ยังไม่ได้กล่าวถึง คือข้อมูลแบบลิสต์(list)

ส่วนกติกาผมได้แสดงให้เห็นว่า โปรแกรมย่อยเป็นรูปแบบหนึ่งของกติกา ตอนต่อไปจะแยกให้เห็นว่า อะไรคือกติกา อะไรคือโปรแกรมย่อยในนัยของโปรล็อก และจะแสดงรูปแบบของการเขียนกติกาแบบอื่นๆ อีก

ท้ายสุดคือ คำถาม ได้แสดงไว้ 2 ลักษณะ แต่ยังมีคำถามอีกลักษณะหนึ่งที่ให้เครื่องถามเรา และเราถามเครื่องว่าแก้ปัญหานี้ได้อย่างไร

เรื่องที่ผมยังไม่กล่าวถึงนี้ จะนำไปกล่าวในตอนต่อไปและต่อๆ ไป หากคุณสนใจในภาษานี้โดยเฉพาะผู้ที่มีความชำนาญพิเศษเฉพาะเรื่อง ขอให้หาโปรแกรมตัวแปรภาษาโปรล็อก (ปกติจะเรียกกันว่า ภาษาโปรล็อกเฉยๆ) ตามร้านจำหน่ายคอมพิวเตอร์ซึ่งจะต้องได้มาพร้อมด้วยคู่มืออีก 1 เล่ม (ถามครั้งแรกไม่มีถามบ่อยๆ คงมี) และถ้าเป็นแบบเดียวกับที่ผมนำมากล่าวก็จะดีครับ